Programmation Exotique
Introduction au PostScript

Maxime Morge morge@di.unipi.it

Licence 2
Université Jean Monnet

2008-2009

Déroulement

- **CM/TD : Mercredi 9h-12h**
  11/02, 18/02, 25/02, 11/03, 18/03, 25/03, 01/04, 22/04, 29/04, 06/05...
- **TP : Lundi 08h-10h** en alternance pour chaque demi-groupe
  16/02, 23/02, 02/03, 09/03, 16/03, 23/03, 30/03, 20/04, 27/04, 04/05...

Déroulement (bis)

- Objectifs
  - Introduction à l'informatique
  - Initiation à un langage de programmation
  - Les types abstraits de données
  - Les fonctions
  - . . . L'art de la programmation
- Contenus
  - Historique, objectifs, méthodes, applications
  - La syntaxe PostScript
  - Les piles et la notation polonaise inversée
  - Définition des fonctions en PostScript
  - La programmation de l'art

Ce que n'est pas l'Informatique

- **Informatique**= science du traitement (automatique) de l'information (par des appareils).
- **Technologies de l'information et de la communication (TIC)**= ens. des techniques utilisées dans les appareils (logiciel + matériel) sont les pièces détachées des appareils.
- *« L'informatique n'est pas plus la science des ordinateurs que l'astronomie n'est celle des télescopes »*. Edsger Dijkstra.
- *« nul ne confond la thermodynamique, la technologie des moteurs à explosion et le mode d'emploi d'un véhicule automobile »*. Bernard Lang.

Ce qu'est l'Informatique

- **Algorithmique**= science des outils et méthodes pour le calcul. Un **algorithme** = ens. d'instruction qui prend des entrées et fournit un résultat.
- La théorie de la calculabilité= étude des limites de ce qui est calculable.
- La théorie de la complexité= étude des resources nécessaires aux algorithmes (temps d'exécution, espace utilisé, ...)
- Cryptographie= Science du secret = cryptographie (écriture secrète) + cryptanalyse (l'analyse du secret).
- La théorie des graphes= science qui étudie les graphes (sommets+ arcs).
- Intelligence artificielle ...

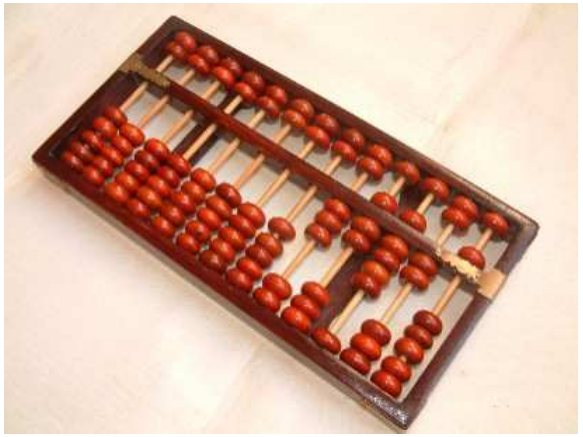
Les outils de la préhistoire

-1750 Le boulier

1642 La pascaline

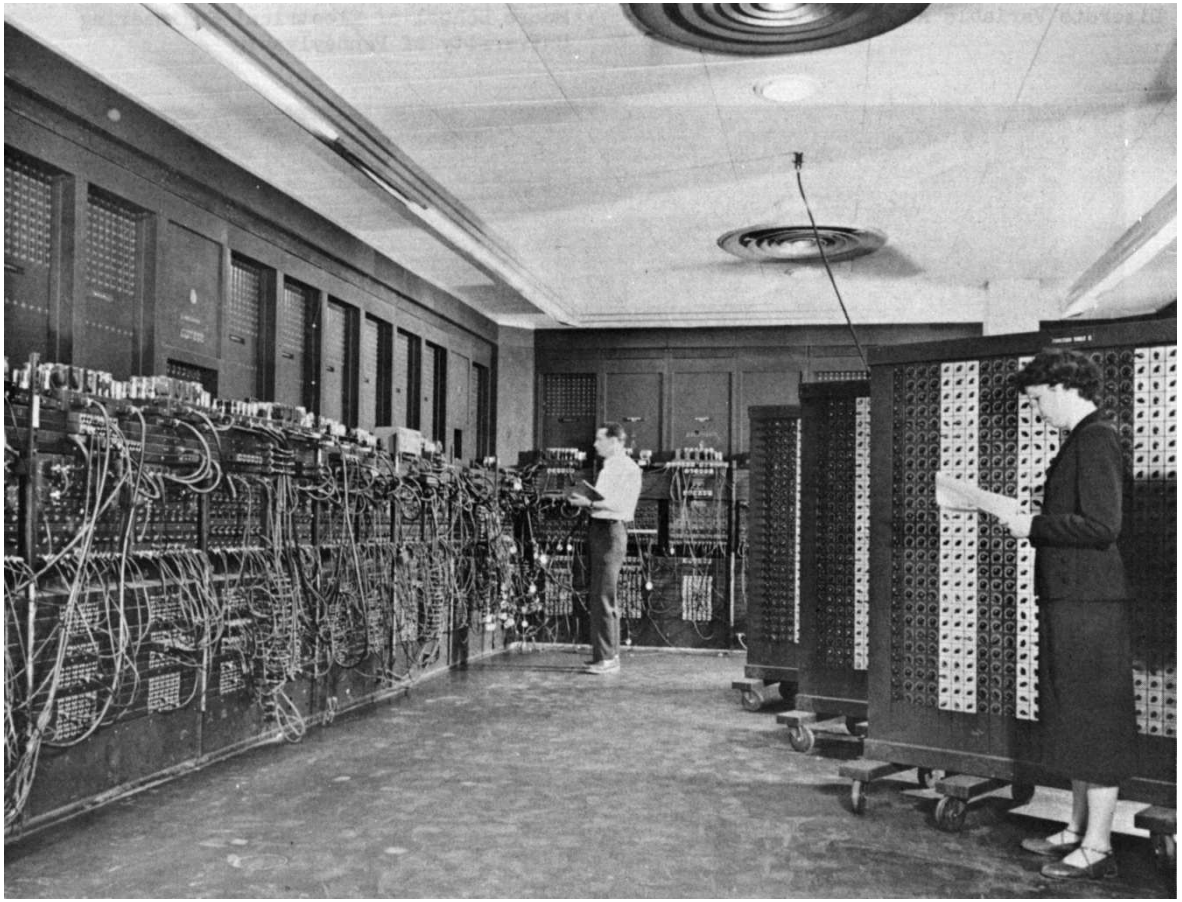
1715 Le métier à tisser

1920 Enigma



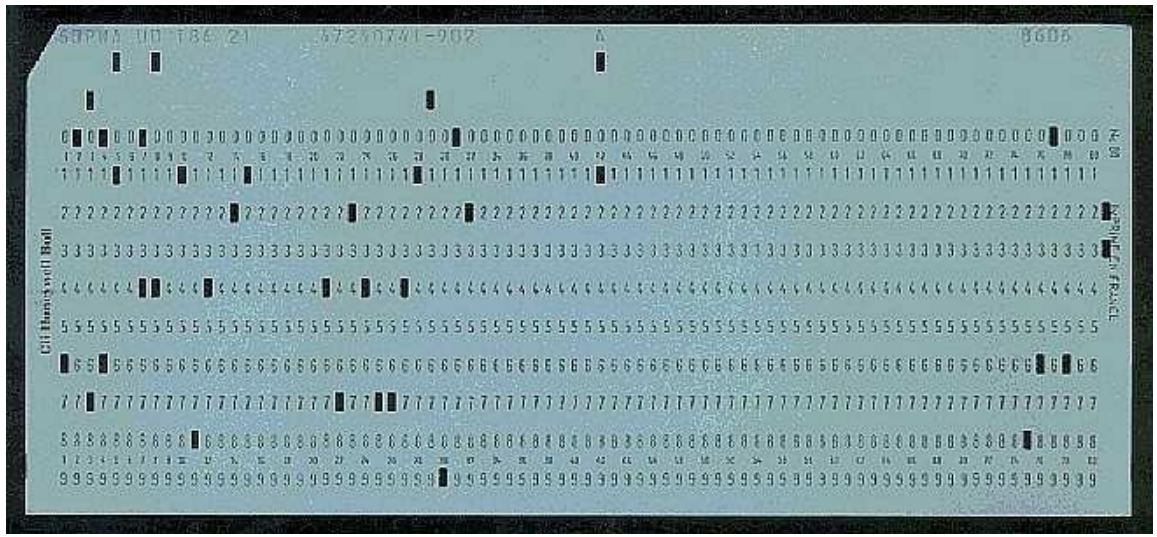
Les calculateurs...

1946 ENIAC (Electronic Numerical Integrator Analyser and Computer)



La première programmeuse

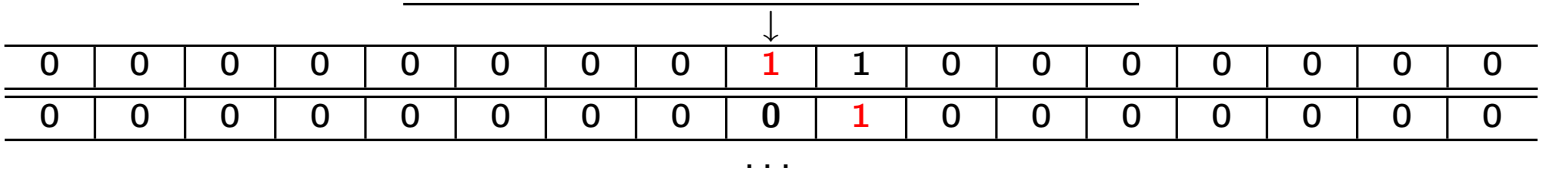
1848 Ada Lovelace créa les premiers programmes (cartes perforées) pour la machine machine analytique de Babbage.



La machine de Turing

1936 Alan Turing propose sa machine.

État	lu	écrit	déplace	état
e1	0	Arrêt		
	1	0	D	e2
e2	0	0	D	e3
	1	1	D	e2
e3	0	1	G	e4
	1	1	D	e3
e4	0	1	G	e5
	1	1	G	e4
e5	0	1	D	e1
	1	1	G	e5



...

La machine de Turing

1936 Alan Turing propose sa machine.

État	lu	écrit	déplace	état
e1	0	Arrêt		
	1	0	D	e2
e2	0	0	D	e3
	1	1	D	e2
e3	0	1	G	e4
	1	1	D	e3
e4	0	1	G	e5
	1	1	G	e4
e5	0	1	D	e1
	1	1	G	e5

↓

0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
								...								
0	0	0	0	0	0	0	0	1	1	0	1	1	0	0	0	0

Les premiers programmes d'ordinateur

- En assembleur

```
.global _start BONJ: .ascii "Bonjour\n" ; Definition en mémoire de
la chaîne à afficher. \n correspond au saut de ligne _start: mov $4
, %eax ; Mettre 4 dans le registre eax (appel système ''Write'')
mov $1
, %ebx ; Mettre 1 dans le registre ebx (descripteur de fichier ''STDOUT'')
mov $BONJ
, %ecx ; Mettre l'adresse mémoire de notre chaîne de caractère dans le registre ecx
mov $8 , %edx ; Mettre la taille de la chaîne dans edx
int $0x80 ; Interruption 0x80, éxécutant un appel système sous
Linux)
 mov $1 , %eax ; Mettre 1 dans eax (appel système ''Exit'')
mov $0 , %ebx ; Mettre 0 dans ebx (valeur de retour du programme)
int $0x80 ; Interruption 0x80, éxécutant un appel système sous
Linux)
```

- **En langage C**

```
#include <stdio.h>
main()
{
    printf("Bonjour\n");
}
```

Un peu de vocabulaire

- Le **langage de programmation**= la syntaxe et la grammaire.
- Le **code source**= ens. d'instructions lisibles, la recette.
- Le **code objet** (binaire ou exécutable)= ens. d'instructions (illisibles) de code machine.
- Le **compilateur**= traducteur d'un langage source vers un langage cible.
- L'**interpréteur**= prog. info. qui analyse, traduit et exécuter un code source.

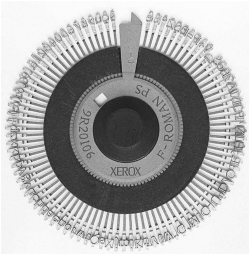
Les paradigmes de programmation

- La programmation **déclarative**, *le quoi*
  - la programmation fonctionnelle (LISP, Scheme, Caml)
  - la programmation logique (Prolog)
- La programmation **impérative**, **le comment** (C, Pascal)
  - la programmation spaghetti
  - la programmation structuré/procédurale
- La programmation **orienté objet** (C++, Java)

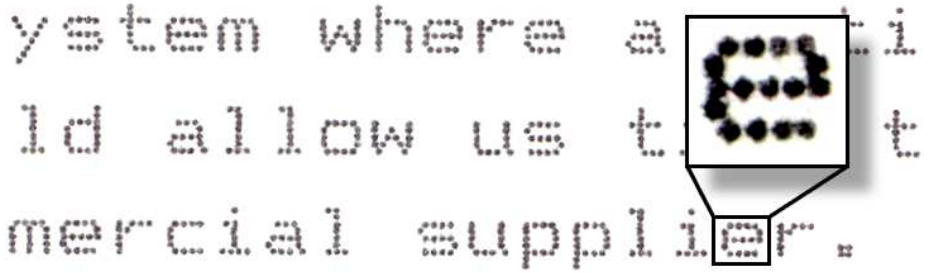
La préhistoire

- Les imprimantes à marteau

- Les imprimantes à marguerite



- Les imprimantes matricielles



Pourquoi PostScript ?

Début 80 L'âge sombre :

- à chaque modèle d'imprimante matricielle son langage ;
- impossible d'échanger des données ;
- les caractères décrit en mode point (*bitmap*) sont de mauvaise qualité.

Début 80 John Warnock et Chuck Geschke développent Interpress à Xerox.

1984 Ils (Adobe) proposent PostScript level 1 :

- indépendant du périphérique ;
- **format ouvert**, les spécification de sont disponibles (n'importe qui peut écrire un programme PostScript)
- **format non libre**, les fabricants achètent une licence ;

1985 Le driver PostScript de l'Apple LaserWriter (7000 US$)

1986 La killing application d'Apple = LaserWriter + PostScript + PageMaker

Et après ?

1991 PostScript Level 2 améliore la vitesse, la fiabilité, filtre d'inclusion d'images JPEG, ...

1993 Création du format PDF (Portable Document Format)= PostScript + fonts + images+ mot clefs + hyperliens + formulaires + films + ...

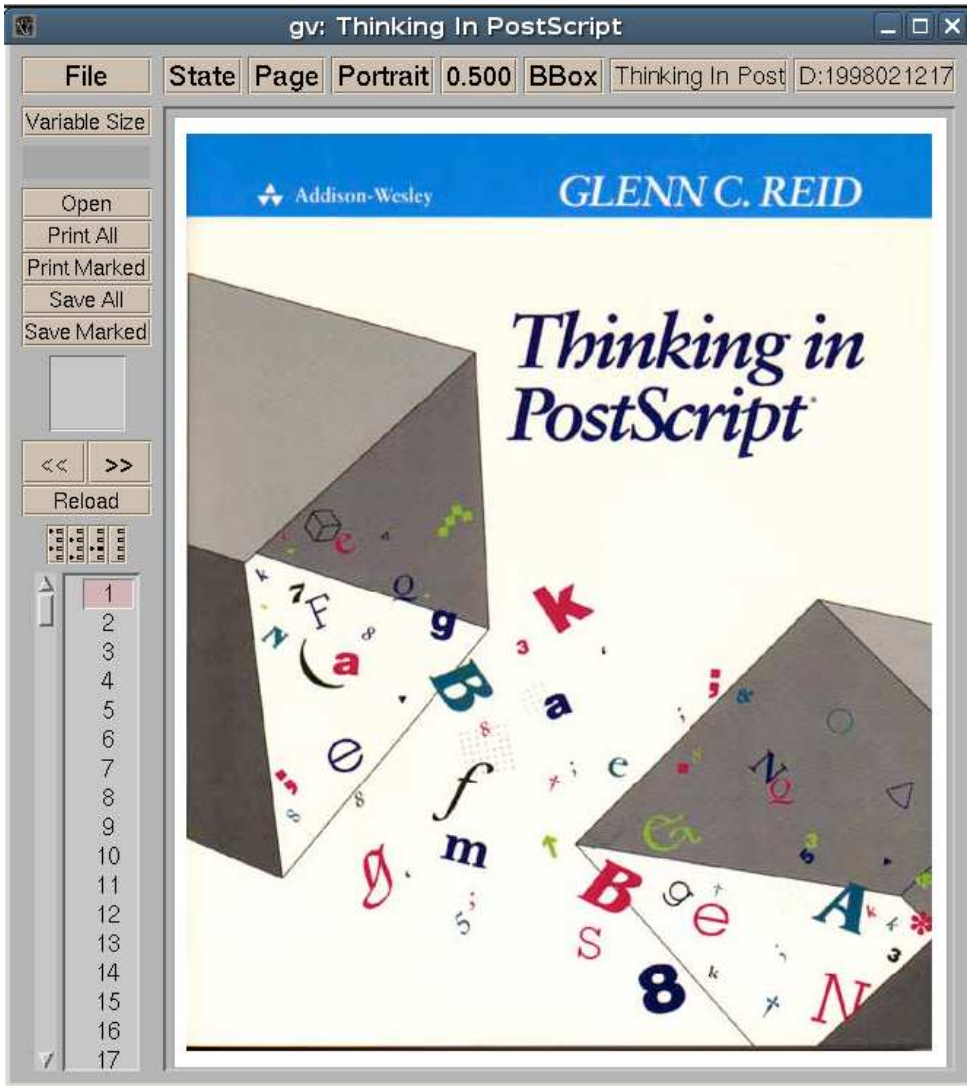
1997 PostScript (Level) 3 améliore la couleur, adjoint de nouveaux filtres

Qu'est ce que le PostScript ?

- Langage de description de page d'impression, i.e. standard indépendant du périphérique.
- Langage interprété, i.e. directement interprétable par la machine (pas de compilateur).
- Langage impératif, i.e. séquence d'instructions.
- Langage procédurale, i.e. contient des routines et sous-routines.
- Langage écrit en ASCII.
- Langage à base de pile, i.e. notation post-fixée (inversée), eg. `1 2 +`

Qu'est ce que GhostScript ?

- Interpréteur multi-plateforme de PostScript et PDF.
- Écrit par Peter Deutsch en 1986 pour le projet GNU.
- GhostView et GV pour les plates-formes à base d'UNIX.
- GSView pour les plates-formes Windows, OS/2 et Linux.
- MacGSView Pour les plates-formes Macintosh.



Qu'est ce qu'un programme PostScript ?

- Une **structure**
  - Le prologue `Prolog`, les définitions utilisées dans le script.
  - Le script `Script`, les procédures définies par le programme.
- Des **objets**= $\langle type, valeur, attribue \rangle$
  - Les objets simples : `operator`, `name`, `int`, `real`, `boo`.
  - Les objets complexes : `array`, `dictionary`, `string`.
  - Les textes : polices de caractère, échelle, angle, ...
  - les figures géométriques, eg `line`, `curves`, `fill colors` ...
  - les images : échelle, orientation, ...

Comment un programme PostScript est interprété ?

4 piles

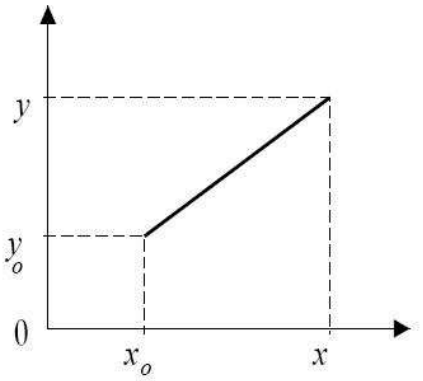
- La piles des opérations pour le calcul.
- La piles du dictionnaire pour les variables/fonctions $\langle cle, valeur \rangle$:
  - celle du système ;
  - celle de l'utilisateur.
- La pile d'éxecution pour les appels de procédures.
- La pile d'état graphique pour sauver l'état courant.

Comment PostScript dessine ?

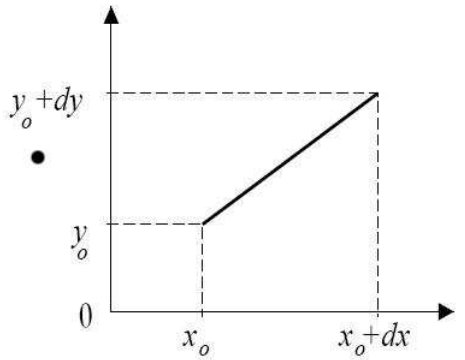
- **Comme nous !**
- La page courante est définie et imprimée.
- Un chemin (*path*), i.e. une succession de points, de lignes et de courbes avec une forme et une position qui peut être :
  - un chemin courant (*current path*) ;
  - un chemin tracé (*stroked path*) ;
  - un chemin rempli (*filled path*) ;
  - un chemin détourré (*clipping path*) qui peut être mis à l'échelle, translaté, tourné ...
- Votre espace de travail= sytème de coordonnée cartésienne dont l'origine est en bas à gauche.

Les commandes de déplacements

- Crayon levé
  - Position absolue
    `x y moveto` $\Rightarrow$ _ déplacement de (x_0, y_0) vers (x, y)
  - Position relative
    `dx dy rmoveto` $\Rightarrow$ _ déplacement de (x_0, y_0) vers $(x_0 + dy, y_0 + dy)$
- Crayon baissé
  - Position absolue
    `x y lineto` $\Rightarrow$ _ déplacement de (x_0, y_0) vers (x, y)
  - Position relative
    `dx dy rlineto` $\Rightarrow$ _ déplacement de (x_0, y_0) vers $(x_0 + dy, y_0 + dy)$



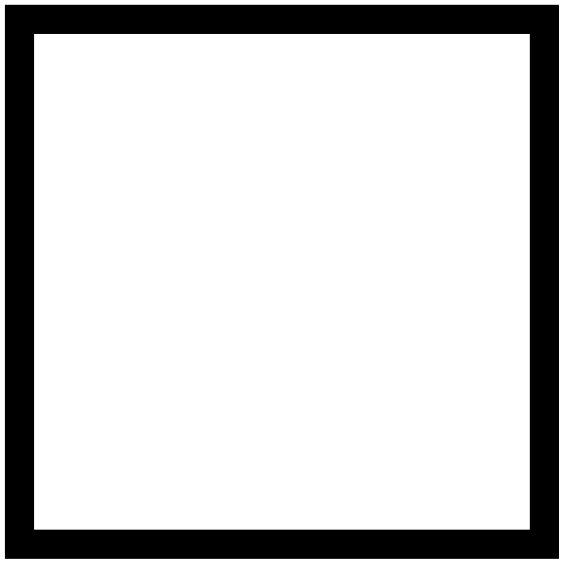
x y **lineto**



dx dy **rlineto**

Les commandes de déplacements (exemples)

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 5 5 105 105
%%BeginProlog
%%EndProlog
newpath
10 10 moveto
90 0 rlineto
0 90 rlineto
-90 0 rlineto
closepath
5 setlinewidth
stroke
showpage
%%EOF
```

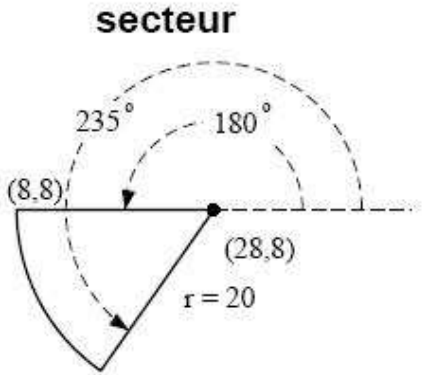


```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 5 5 105 105
%%BeginProlog
%%EndProlog
newpath
10 10 moveto
90 0 rlineto
0 90 rlineto
-90 0 rlineto
closepath
.5 setgray
fill
showpage
%%EOF
```

Les commandes d'arc de cercle

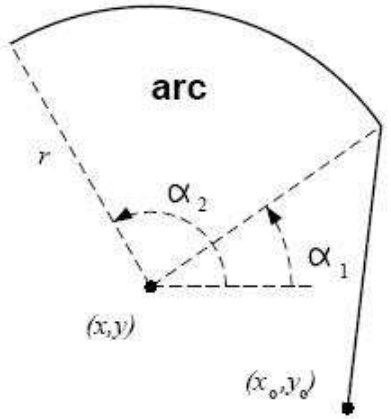
- `x y r` α_1 α_2 `arc` $\Rightarrow$ `_`
  arc de cercle de centre (x, y) et de rayon r commençant à l'angle α_1 jusqu'à l'angle α_2.

```
8 8 moveto
28 8 20 180 235 arc
```



- `x y r` α_1 α_2 `arcn` $\Rightarrow$ `_`
  idem + segment entre le point courant et l'arc.

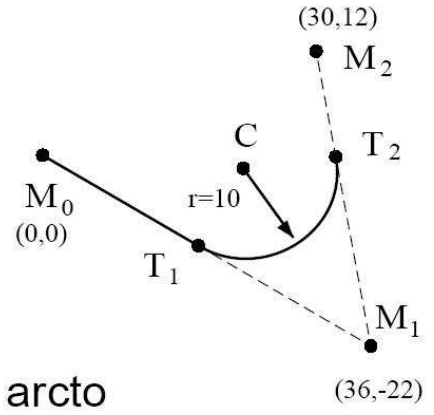
```
15 0 moveto
0 0 30 0 55 arc 0 0
15 55 0 arcn
```



Les commandes d'arc de cercle (suite)

- x_1 y_1 x_2 y_2 r `arcto` $\Rightarrow$
  x_{t1} y_{t_1} x_{t_2} y_{t2}
  trace un segement du point courant vers M_1 (x_1, y_1) qui se termine en arc de cercle de rayon r qui se termine par un point sur le segment entre M_1 et M_2 (x_2, y_2) + retourne les deux points de l'arc.

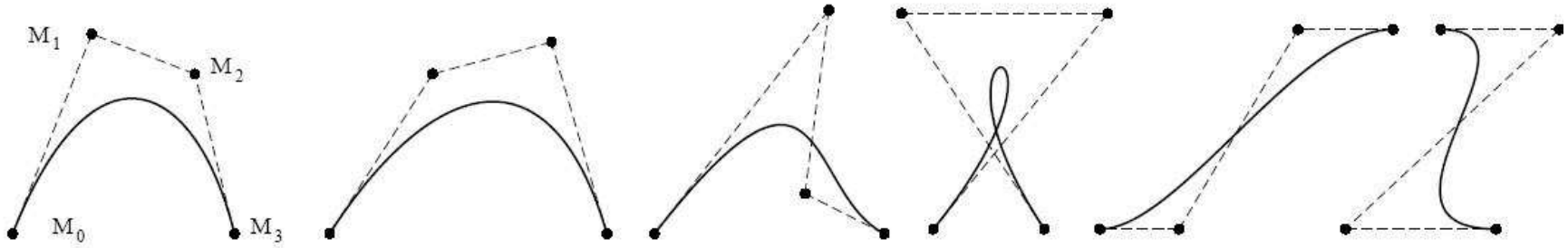
```
0 0 moveto
36 -22 30 12 10 arcto
```



Les commandes de courbe

Les courbes s'appuyent sur 4 points dont le premier et le point courant.

- x_1 y_1 x_2 y_2 x_3 y_3 `curveto` $\Rightarrow$ _
- dx_1 dy_1 dx_2 dy_2 dx_3 dy_3 `rcurveto` $\Rightarrow$ _



Tracage et remplissage

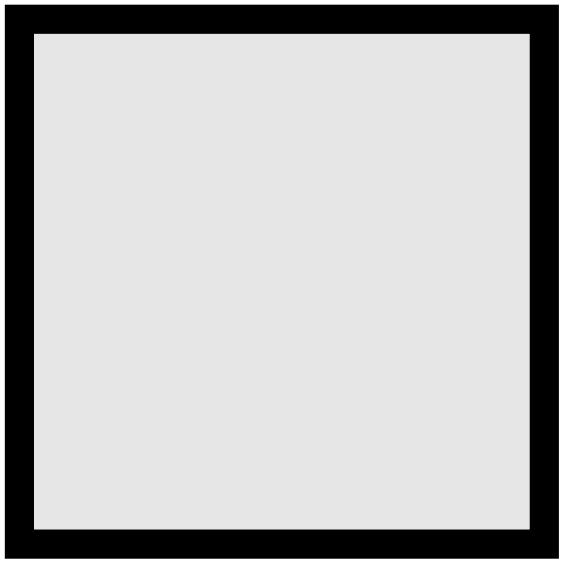
- `_ stroke` ⇒ `_`
  convertit le chemin (ligne théorique) en ligne visible
- `_ fill` ⇒ `_`
  remplit en grisé la surface délimitée par un chemin
- `x setgray` ⇒ `_`
  nuance la couleur de tracage/remplissage ($x \in [0, 1]$)
- `x setlinewidth` ⇒ `_`
  définit la largeur de la ligne

Comment à la fois tracer la ligne et remplir la surface d'un chemin ?

- `gsave` ⇒ `_`
  sauvegarde l'état de la pile graphique
- `grestore` ⇒ `_`
  rappele l'état de la pile graphique

Tracage et remplissage (exemple)

```
...
closepath
gsave
0.9 setgray
fill
grestore
5 setlinewidth
stroke
showpage
%% EOF
```



Les filtres

- `_ clip` ⇒ `_`
  interdit aux fonctions d'encrage `fill` et `stroke` d'agire en dehors de la zone
- `_ initclip` ⇒ `_`
  revient au filtre d'origine, la totalité de la page.

Les chemins en résumé

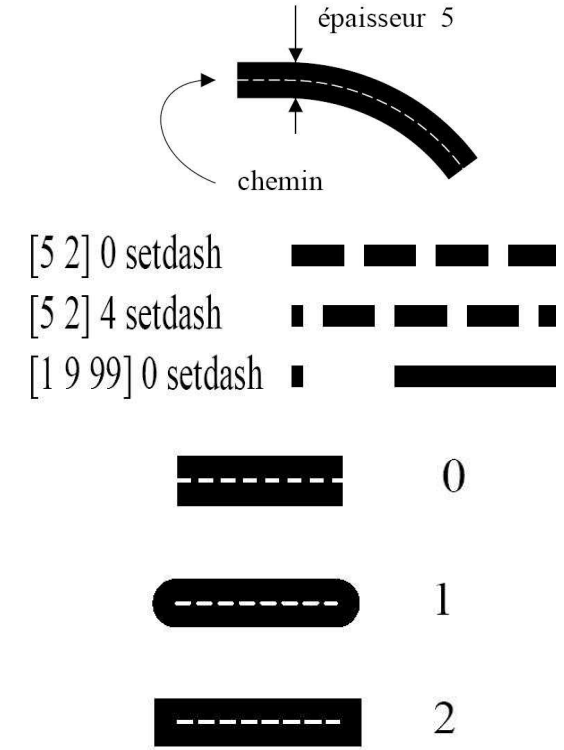
1. Définir un chemin (`newpath`).
2. Décrire un chemin en :
   - donnant une succesion de points (`lineto|moveto`) ;
   - indiquant la nature des traits (`arcto|curveto`) ;
   - specifiant les caratéristiques graphiques (`setgray|setlinewidth`).
3. Matérialiser ce chemin (`fill|stroke|clip`).
4. Décrire un nouveau chemin en ...
5. ...
6. Imprimer (`showpage`).

_ `currentpoint` $\Rightarrow$ *xy* _

Met les coordonnées du point courant dans la pile utilisateur.

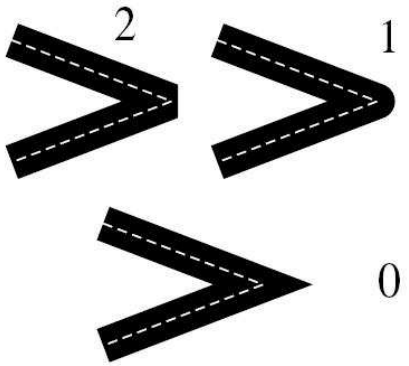
Caractéristiques graphiques : les traits

- `x setlinewidth` ⇒ _
  épaisseur du trait
- $[n_1 n_2 \ldots]$ m `setdash` ⇒ _
  motif du pointillé avec :
  - n_1, la longueur du trait noir ;
  - n_2, la longueur de l'espace ;
  - m, le décalage.



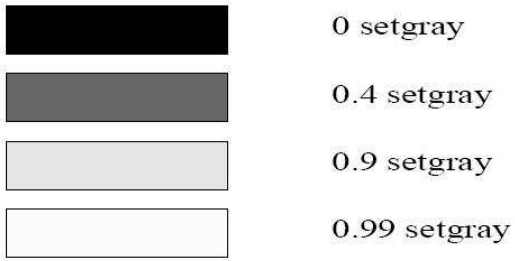
Caractéristiques graphiques : les traits

- *n* `setlinejoin` ⇒ _
  l'angle entre lignes avec
  - $n = 0$, aiguë ;
  - $n = 1$, arrondie au rayon $x/2$;
  - $n = 2$, chanfreinée.



Caractéristiques graphiques : les couleurs

- *g* `setgray` ⇒ _
  le niveau de gris entre 0 (le noir) et 1 (le blanc).
- *r v b* `setrgbcolor` ⇒ _
  la couleur est une combinaison de rouge (*r*), de vert (*v*) et de bleu (*b*).
- *h s b* `sethsbcolor` ⇒ _
  la couleur a pour teinte *h* (hue), pour pureté *s* (saturation) et pour intensité *b* (brightness).



À emporter

- **Objectifs :**
  - Introduction à l'informatique.
  - Initiation à un langage de programmation.
- **Contenu :**
  - Historique, objectifs, méthodes, applications.
  - La syntaxe PostScript.
- **Perspectives :**
  - Les Types Abstrait de Données.
  - Les piles PostScript et la notation polonaise inverse.